Welcome to
Introduction to Computer Science I
CSE 1020

moodle.yorku.ca

Instructor

- Name: Franck van Breugel
- Email: franck@cse.yorku.ca
  Please use your EECS or York account to send me email
- Office: Lassonde Building, room 3046
- Office hours: Mondays, 13:00-14:00, and Wednesdays and Fridays, 10:30-11:20 in the office, and Fridays, 14:30-15:45 in the lab, or by appointment

Evaluation

- Weekly: programming exercises (0%)
- January 17: test (10%)
- January 24: test (10%)
- January 31: test (10%)
- February 7: test (10%)
- February 28: test (10%)
- March 21: test (10%)
- April 4: test (10%)
- Exam period: final exam (30%)
- Some bonus marks.

Each test will be 45 minutes and will consist of one programming question and several conceptual questions.

The final exam will be 120 minutes and will consist mainly of conceptual questions. The exam period is April 8–24.

Weekly programming exercises

- Each week (week 1–11), there is one programming exercise (see Moodle).
- Although the programming exercises are worth 0%, students that successfully submit a programming exercise will receive feedback.
- The programming questions of the tests are similar to the programming exercises.
- The programming exercises have to be completed before Sunday (for example, the programming exercise of week 1 has to be completed on January 11 at the latest).

Programming

Practice, practice, practice, . . .

- weekly programming exercises
- programming exercises in the textbook

Labs

- Lassonde Building, room 1002 and 1006
- Fridays, 14:30-16:00
- The first lab is held this week.
- Tests will be held during the labs.

Activate your EECS account:

`www.eecs.yorku.ca/activ8`.

Do this *before* Friday (it takes roughly 25 minutes to activate your account).

Drop deadline

March 7

Until this date you can drop the course without getting a grade for it and, hence, it will not affect your gpa.

www.registrar.yorku.ca/enrol/dates/fw13.htm contains important dates.

Academic honesty

"If you put your name on something, then it is your work, unless you explicitly say that it is not."

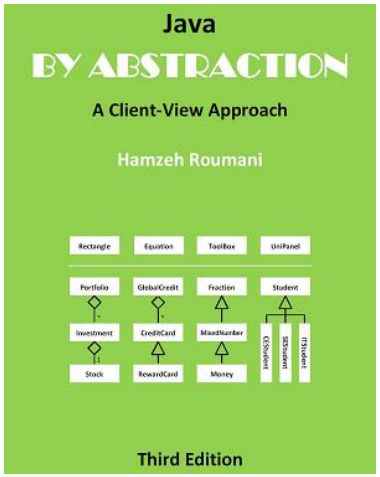
www.cse.yorku.ca/admin/coscOnAcadHonesty.html contains more details.

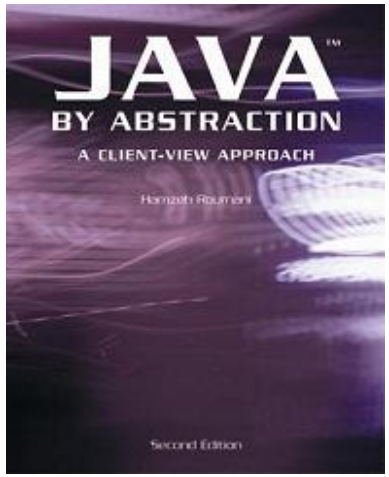
Textbook

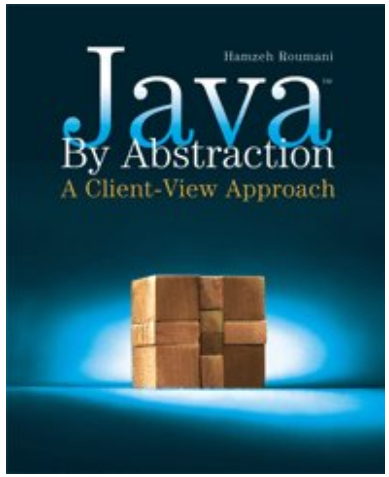
Hamzeh Roumani. *Java by Abstraction: A Client-View Approach*. First or second or third edition. Pearson, Toronto. 2005 and 2007 and 2010.

Why this textbook?

- As far as I know, this is the only textbook that uses the client-view (more about this view later in the course).
- The author is an award winning lecturer and teaches at York.







Study the textbook

The textbook is required for this course. It is on reserve in the Steacie library. Second-hand copies are available.

Studying only the slides and your lecture notes is *not* sufficient. There will be questions on tests about material that is *not* covered in class. Therefore, you should study the textbook.

Although you need to memorize some material, most of the material you have to understand.

Textbook

Each chapter consists of

- reading material and contains several types of "boxes"
  - Programming Tip: essential
  - Java Details: useful
  - In More Depth: read and try to understand (don't worry if you don't understand them completely)
- review questions (try some yourself after having read a chapter),
- a lab (do it yourself),
- exercises (try some yourself, good preparation for tests), and
- eChecks.

Advice

Apart from attending the lectures and the labs, a student will need to spend on average another 6 hours per week for studying the textbook and practicing writing code.

If you have questions

- ask them during the lab, before/during/after the lecture
- come to my office hours
- send me email (from your EECS or York account)

Expectations ...

... at University are different from expectations at high school.

Advise: keep up with the material, studying only the day before the test is *not* sufficient.

In general, half of the students in this course do not complete it or fail the course. Let's try to do better this term.

How does this course fit within the program?

- CSE 1020: how to use classes
- CSE 1030: how to implement classes
- CSE 2011: how to represent and manipulate data

The programming skills and knowledge of basic concepts that you develop in these three courses will be useful in many other courses.

However . . .

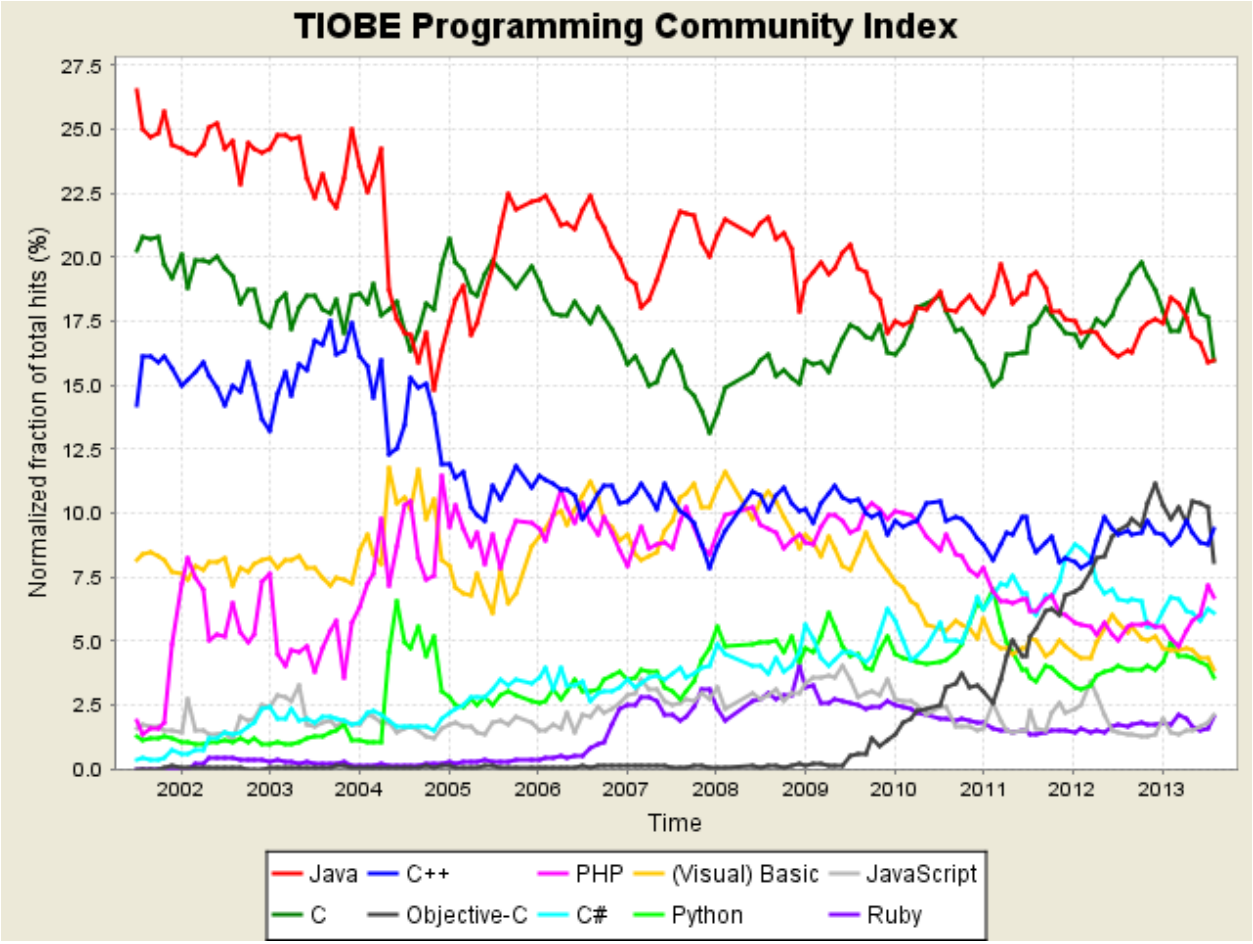
computer science $\neq$ programming

Programming languages

ABC, Ada, Algol 60, Algol 68, Alice, APL, Basic, BPEL, C, C++, C#, Caml, COBOL, CSP, Eiffel, Emacs Lisp, Erlang, Esterel, Fortran, Haskell, IMP, Java, JavaScript, LaTeX, Lisp, LOTUS, Lustre, Maple, Mathematica, MATLAB, Mesa, Metafont, Miranda, ML, Modula-2, Oak, Oberon, Objective Caml, occam, Pascal, Perl, PHP, Pict, Pizza, PL/I, PostScript, Prolog, Promela, Python, Scala, Scheme, Simula, Smalltalk, SNOBOL, SOAP, Tcl, TeX, Turing, Visual Basic, Z, ...

Why Java?

Java is well-designed and popular.



source: `www.tiobe.com`

Why Java?

Because Java is popular, there are

- many textbooks about Java,
- many web pages about Java, and
- many software packages written in Java.

However, . . .

. . . this is *not* a course about Java.

Java is used to introduce you to programming and to explain several fundamental concepts.

In other courses you will get familiar with other languages.

- CSE 2031: C
- CSE 2041: Javascript
- CSE 3401: Prolog
- CSE 3311: Eiffel

Oak

In the early 1990s, James Gosling and some of his colleagues at Sun Microsystems, developed a programming language to program device controllers. The language was called Oak after an oak tree that stood outside Gosling's office. The language was expanded to a general purpose programming language and was renamed Java (because Oak was already a trademark). On May 23, 1995, Java was announced.

"Even though the Web had been around for 20 years or so, with FTP and telnet, it was difficult to use. Then Mosaic came out in 1993 as an easy-to-use front end to the Web, and that revolutionized people's perceptions. The Internet was being transformed into exactly the network that we had been trying to convince the cable companies they ought to be building. All the stuff we had wanted to do, in generalities, fit perfectly with the way applications were written, delivered, and used on the Internet. It was just an incredible accident. And it was patently obvious that the Internet and Java were a match made in heaven. So that's what we did."

James Gosling

James Gosling

James Gosling was born near Calgary in 1955. He received his BSc in computer science from the University of Calgary and his PhD from Carnegie Mellon University. In 2007, he was made an officer of the Order of Canada. He is best known as "the father of the Java programming language."

Trivial problem

Write a Java program that prints the Java's age.

Step 1

Solve the problem.

Step 2

Write the program.[1]

For this you need an editor:

- jEdit
- eclipse
- NetBeans
- Notepad
- ...

[1] Instead of program, we often call it an app(lication).

Name of a Java application

According to the Java Language Specification, the name of an application should be a sequence of letters and digits and the symbols _ and $, starting with a letter.

Code conventions

Rules how to write your code such as

- naming conventions for applications, variables, etc,
- indentation,
- etc.

Appendix C of the textbook contains the code conventions to which you and I will (try) adhere during this course. (We will not adhere to the rule for placement of braces.)

Another example of code conventions can be found at the URL babelfish.arc.nasa.gov/trac/jpf/wiki/devel/coding_conventions.

Code convention for application names

- Capitalize first letter.
- If the name is made up of more than one word, the capitalize the first letter of each.
- If the name is an acronym, then capitalize all letters.

Question

Which of the following class names adhere to the code convention?

- URL
- Sequence_Of_Characters
- Url
- CharacterSequence

Class header

```
public class AgeOfJava {
    public  static  void main(String[]  args)  {

    }
}
```

Class body

```
public class AgeOfJava {
    public static void main(String[] args) {

    }
}
```

Method header

```
public class AgeOfJava {
    public static void main(String[] args) {

    }
}
```

Method body

```
public class AgeOfJava {
    public static void main(String[] args) {
        ...
    }
}
```

Keywords

```
public class AgeOfJava {
    public static void main(String[] args) {

    }
}
```

Identifiers

```
public class AgeOfJava {
    public static void main(String[] args) {

    }
}
```

Separators

```
public class AgeOfJava {
    public static void main(String[] args) {

    }
}
```

Another code convention

Do not use magic numbers in expressions.

Magic numbers: numbers different from 0, 1, -1, 2, -2

Why should we not use magic numbers in expressions?

- Obscures the intent of the numbers
  ((100 * 60 * 60) / 9.58) / 1609.34
- Increases opportunities for subtle errors
- Makes it more difficult for the code to be adapted and extended in the future

Code convention for variables names

- Use lowercase letters.
- If the name is made up of more than one word, capitalize the first letter of each subsequent word.

Question

Which of the following variable names adhere to the code convention?

- myVariable
- MyVariable
- my_variable

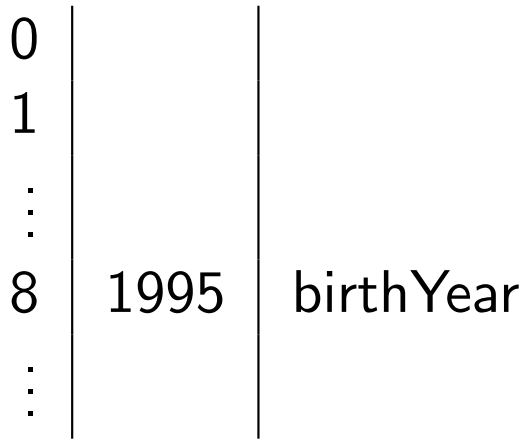
Memory model

int birthYear = 1995;

0		
1		
⋮		
8	00000000	birthYear
9	00000000	
10	00000111	
11	11001011	
⋮		

Memory model

int birthYear = 1995;



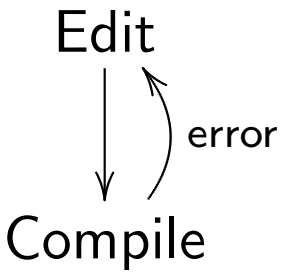
Terminology

System.out.println(age);

- System is (the name of) a class
- out is (the name of) a field
- println is (the name of) a method

Step 3

Compile the app: javac AgeOfJava.java

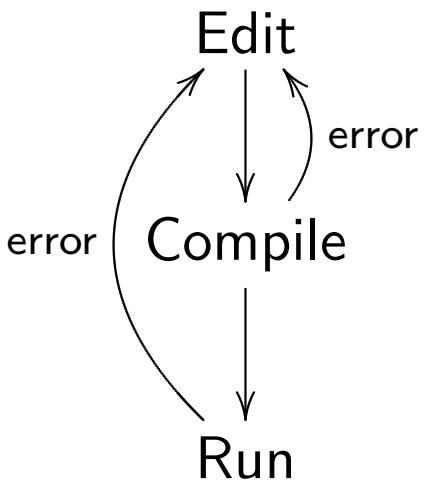


CSE 4302: Compilers and Interpreters

eclipse automatically compiles the app.

Step 4

Run the app: java AgeOfJava



In eclipse, you press the button with the green circle and the white arrow.

Nontrivial problem

By the end of this course you should be able to develop an app that

- randomly picks ten cities from a list of Ontarian cities,
- extracts the current temperature for each city from `weather.gc.ca`, and
- displays the temperatures.

The four steps

1. Solve the problem.
2. Write the app (using an editor such as eclipse).
3. Compile the app (done automatically in eclipse).
4. Run the app (pressing a button in eclipse).

To do

Do the following before Wednesday.

- Read general material at Moodle.
- Study pages 1–16 and Appendix C of the textbook.
- Activate your EECS account: www.eecs.yorku.ca/activ8.

One bonus mark for the first student who emails me a picture of someone studying the textbook for this course on public transport (bus, subway, streetcar, train, . . .).